

Assegnamento 2: Invasion Percolation

AA 2015/16

Invasion percolation è una semplice forma frattale basata su un modello di diffusione di fluidi. In due dimensioni, l'invasion percolation si simula come segue.

Passo 1 Si genera una griglia rettangolare $n \times m$ di numeri casuali nell'intervallo $[1, r]$. Ad esempio, una griglia 5×5 di numeri nel range $[1.0, 10.0]$ può essere la seguente:

```
2.6 1.7 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 1.2 2.9 7.7
6.1 2.6 9.0 3.5 1.1
8.3 8.4 1.8 1.9 5.2
```

Passo 2 si marca la posizione del centro (posizione $\lfloor n/2 \rfloor, \lfloor m/2 \rfloor$) della griglia come *pieno*, ad esempio con (***):

```
2.6 1.7 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 *** 2.9 7.7
6.1 2.6 9.0 3.5 1.1
8.3 8.4 1.8 1.9 5.2
```

Passo 3 si effettua un ciclo in cui ad ogni passo:

1. si esaminano i valori adiacenti a tutte le celle già invase (marcate con ***) nel nostro esempio). Nel calcolo dei valori adiacenti ai valori invasi la matrice deve essere considerata *toroidale*, ovvero la riga $n-1$ è adiacente alla riga 0 e la colonna $m-1$ è adiacente alla colonna 0,
2. si sceglie fra tutti i valori adiacenti (non ancora marcati) la cella con il valore minore (se il minimo occorre in più celle se ne sceglie una qualsiasi),
3. si marca tale cella come *invasa*.

Si vuole visualizzare l'evoluzione della diffusione del liquido mettendo in evidenza ad ogni passo l'ultima cella invasa. Questo puo' essere fatto contrassegnando tale cella con il +++.

Ad esempio, visualizziamo l'evoluzione della matrice iniziale del passo 2 :

```
2.6 1.7 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 +++ 2.9 7.7
6.1 2.6 9.0 3.5 1.1
8.3 8.4 1.8 1.9 5.2
```

Nel primo passo gli 8 vicini della cella (2,2) sono:

```
3.8 3.9 9.2
2.9 +++ 2.9
2.6 9.0 3.5
```

L'evoluzione della griglia è la seguente: alla prima iterazione del passo 3 si ha (+++ indica la cella selezionata ad ogni nuovo passo)

```
2.6 1.7 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 *** 2.9 7.7
6.1 +++ 9.0 3.5 1.1
8.3 8.4 1.8 1.9 5.2
```

alla seconda iterazione

```
2.6 1.7 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 *** 2.9 7.7
6.1 *** 9.0 3.5 1.1
8.3 8.4 +++ 1.9 5.2
```

alla terza

```
2.6 +++ 7.2 4.5 3.8
2.0 3.8 3.9 9.2 3.8
4.4 2.9 *** 2.9 7.7
6.1 *** 9.0 3.5 1.1
8.3 8.4 *** 1.9 5.2
```

alla quarta

```
2.6 *** 7.2 4.5 3.8
1.0 3.8 3.9 9.2 3.8
4.4 2.9 *** 2.9 7.7
6.1 *** 9.0 3.5 1.1
8.3 8.4 *** +++ 5.2
```

e così via... Si richiede di implementare l'insieme di funzioni C il cui prototipo è definito nel file `percolation.h`.

La matrice toroidale della simulazione è rappresentata da una matrice di `double` bidimensionale ed è memorizzata in un array di puntatori a righe (quindi il tipo dell'array è `double **`). Gli elementi della matrice contengono il valore della permeabilità (se non sono state ancora invase) oppure il valore speciale -2 (se è stata appena invasa) e il valore speciale -1 (se è stata invasa ad un'iterazione precedente). Sono fornite dai docenti (file `percolation_docenti.c`) le seguenti funzioni:

- `new_matrix()` che alloca una nuova matrice `n` per `m` sullo heap e ne restituisce il puntatore,
- `print_matrix()` che stampa su uno stream di output (già aperto) la matrice in formato testo

Devono essere invece realizzate dallo studente le funzioni:

```
void free_matrix (double *** pm, unsigned n);
void prettyprint_matrix (FILE * f, double ** a, unsigned n, unsigned m);
int load_matrix (FILE * f, double ** a, unsigned n, unsigned m);
int init_percolation (double ** mat, unsigned n, unsigned m, double a, double b);
int first_step (double ** mat, unsigned n, unsigned m);
int step (double ** mat, unsigned n, unsigned m, double a, double b);
```

In particolare, `free_matrix()` libera la memoria occupata da una matrice, `prettyprint_matrix` stampa la matrice in ASCII art (vedi https://it.wikipedia.org/wiki/ASCII_art per alcune idee) secondo un formato a scelta dello studente, `load_matrix` scrive la matrice con i valori di permeabilità presenti su un file, `init_matrix` inizializza i valori di permeabilità di una matrice in modo casuale, `first_step` inserisce il valore -2 relativo al primo step della simulazione e `step` trasforma il valore -2 in -1 e inserisce il nuovo valore -2 di un generico step. Le caratteristiche delle varie funzioni, i parametri ed i valori da restituire sono specificati nel file `percolation.h`. Le funzioni devono essere tutte realizzate in un unico file `percolation.c`.

I file `test_one.c`, `test_two.c` e `test_three.c` contengono dei test sulle funzioni sviluppate. Tali test possono essere attivati automaticamente utilizzando il `Makefile` come specificato nel file `README`. Solo il codice che supera con successo questi test può essere consegnato.